Estrategia Guiada por Modelos para incluir Aspectos de Seguridad en Sistemas Empotrados Basados en Servicios Web

Juan Pedro Silva Gallino , Miguel de Miguel, Javier F. Briones, Alejandro Alonso

Resumen

En los sistemas distribuidos modernos, como la Internet o Web de las Cosas, la seguridad juega un papel preponderante. Debe prestarse especial atención a la consideración de estos aspectos en las primeras etapas de desarrollo. En este contexto, el desarrollo guiado por modelos de requisitos no funcionales (NF) presenta especial interés, ya que aborda dichas características NF en la etapa de diseño, cuando todavía se pueden realizar análisis, y aún hay margen para modificaciones antes de que éstas sean muy costosas. El uso de estas metodologías guiadas por modelos ofrece beneficios tales como el aumento de la productividad, una mayor reutilización de los elementos de diseño, o una mejor mantenibilidad del sistema. Este artículo presenta una estrategia de desarrollo que permite integrar aspectos NF de seguridad (confidencialidad, integridad, y control de acceso) en los sistemas de software empotrado.

Palabras Clave: Desarrollo Guiado por Modelos, Perfil de Dispositivos para Servicios Web, Políticas de Servicios Web, Seguridad en Servicios Web

1. Introduction

Como indica [Eby (2007)], existe un creciente interés en considerar la seguridad de los dispositivos empotrados, dado su creciente uso en sistemas distribuidos que hacen uso de redes no seguras. Actualmente, los dispositivos de software empotrado con capacidad de ser conectados cuentan cada vez con mayores recursos. Esto lleva a una visión de la Internet de las Cosas (*Internet of Things*, IoT), donde los estándares WS-* son aplicados para obtener la Web de las Cosas (*Web of Things*, WoT). En la WoT, para dispositivos de pocos recursos, o sistemas donde la seguridad no es determinante, existen técnicas más simples para su implementación, por ejemplo los servicios web *REpresentational State Transfer* (REST). Sin embargo, es una opinión compartida entre los desarrolladores [Guinard et al. (2011)] que, para sistemas donde la seguridad juega un papel preponderante, el uso de lo que se conoce como *Big Web Services* (que hacen uso de los estándares WS-*) se convierte en fundamental. Esto permite la utilización de modernas técnicas de seguridad extremo a extremo, como las definidas por *WS-Security*, que pueden ser empleadas para brindar a estos dispositivos la seguridad que los sistemas necesitan, facilitando la compatibilidad cuando dichos dispositivos deben interactuar con otros servicios no empotrados. Teniendo en cuenta lo limitado de los recursos de estos dispositivos, se ha definido el estándar OASIS *Devices Profile for Web Services* (DPWS) [OASIS (2009)], el cuál define una serie de pautas que deben ser seguidas en el aspecto de la seguridad.

El uso del desarrollo guiado por modelos ya se ha mostrado efectivo en el diseño y desarrollo de sistemas empotrados, mientras que la seguridad también ha sido estudiada como caso de aplicación de dichas técnicas. Sin embargo, en la actualidad en general el diseño de la seguridad hace uso de lenguajes monolíticos específicos para dicho fin, o de lenguajes de modelado genéricos, pero menos expresivos. La primer aproximación no es demasiado flexible a la hora de incorporar nuevas tecnologías, o incluso otras características NF que no han sido consideradas al momento de definir el lenguaje. La segunda, por otra parte, no ofrece el mismo poder de expresividad y cercanía con el dominio del problema.

Este artículo presenta la aplicación de una estrategia de desarrollo, técnicas de modelado, y herramientas que permitan integrar aspectos NF de seguridad, a los sistemas de servicios desarrollados mediante dispositivos de software empotrado. Se buscan, en particular, cuatro características deseables en estos elementos:

- Que ofrezcan mecanismos de modelado apropiados al nivel de abstracción en el cual el desarrollador se encuentra

modelando.

- Que minimize el conocimiento técnico NF requerido a los desarrolladores participantes.
- Que ofrezca un nivel de modularidad que permita, entre otras cosas, potenciar la reutilización, la mantenibilidad, y la privacidad del diseño.
- Finalmente, que ofrezca medios apropiados para la realización de análisis en etapas tempranas del desarrollo.

El proceso que se presenta, que extiende y detalla el presentado por primera vez en [Silva Gallino, J. P. and de Miguel, M. A. and Briones, J. F. and Alonso, A. (2011a)], hace uso de ontologías para el descubrimiento de modelos NF almacenados en repositorios. A continuación, dichos modelos son utilizados en una composición de modelos, y posterior generación de un modelo completo de diseño de sistemas con conciencia de la seguridad. La aproximación propuesta integra aspectos NF de seguridad en el desarrollo de sistemas empotrados distribuidos. Partiendo de descripciones de actividad de alto nivel, el proceso incorpora elementos técnicos a través de una cadena de transformaciones y composiciones, hasta llegar a una implementación final (código y configuración) de los dispositivos empotrados individuales con conciencia de la seguridad.

El artículo se estructura como sigue. La Sección 2 describe la estrategia presentada, y en particular, la sección 2.1 describe los pasos del proceso de desarrollo propuesto, detallando el entorno de desarrollo en 2.1. La Sección 3 presenta un caso de uso en torno al control de velocidad de un aerogenerador, y detallando su implementación a lo largo de los pasos propuestos anteriormente. Dentro de esta sección, se realiza una breve introducción al perfil DPWS de servicios web (WS) para dispositivos en la sección 3.4. El trabajo relacionado más relevante se presenta en la sección 4. Finalmente, la sección 5 ofrece algunas conclusiones y posibles líneas de trabajo futuras.

2. Integrando Características No Funcionales en Sistemas Orientados a Servicios

En este artículo, se define una estrategia para la incorporación de aspectos NF en dispositivos empotrados distribuidos. En particular, la solución está enfocada a características NF proclives a ser descritas como políticas, de las cuales la seguridad y el control de acceso (CA) son ejemplos prominentes. Esta sección describe el proceso, aplicado al caso particular de las características de seguridad.

Un elemento a considerar es el conocimiento técnico necesario para expresar correctamente (tanto sintáctica como conceptualmente) configuraciones de seguridad. En sistemas distribuidos, los procesos y colaboraciones de los diferentes dispositivos puede representarse a alto nivel mediante diagramas de flujo o de actividad. Los diseñadores que participen a este nivel pueden tener una idea abstracta de los requisitos de seguridad necesarios para dicho sistema, pero probablemente no cuenten (o tal vez sí, pero no necesariamente) con los conocimientos técnicos necesarios para expresar una completa configuración de seguridad que satisfaga dichos requisitos, o con la información referida a la disponibilidad de infrastructura, mecanismos, o algoritmos de seguridad, por ejemplo.

Durante la participación en proyectos de investigación previos [CDTI (2006)], surgió la necesidad de estudiar las características NF de los sistemas de forma independiente de los aspectos funcionales, para promover la reutilización del diseño, tanto del funcional como del NF. En este caso, los dispositivos empotrados pueden ser utilizados en diferentes entornos, con requisitos de seguridad cambiantes, o en entornos de similares características, pero realizando tareas funcionales diferentes.

Esta heterogeneidad implica que las características NF de los sistemas deben soportarse en diferentes maneras y formas. Las aproximaciones propuestas por *Multi-Dimensional Separation of Concerns* (MDSOC) [Tarr et al. (1999)], y reutilizadas por *Aspect-Oriented modeling* (AOM) [Elrad et al. (2002)] son enfoques candidatos a solucionar este factor. Estos enfoques comparten ideas comunes, que permiten la separación de aspectos funcionales y NF de los sistemas *durante las etapas de diseño*, y representan la base sobre la cual se apoya el trabajo presentado en este artículo.

2.1. *Proceso de Desarrollo*

El proceso de desarrollo que se ha definido está compuesto por siete pasos, que pueden visualizarse en la Figura 1, y que se describen a continuación:

1. La entrada inicial del proceso está compuesta de un diagrama de procesos (un diagrama BPMN (*Business Process Modeling Notation* [OMG (2011)]) o, alternativamente, un diagrama UML (*Unified Modeling Language*) de actividad), anotado con primitivas abstractas de seguridad.
2. Estas primitivas a nivel CIM (*Computation Independent Models*), para las cuales se identifíca correspondencias en una ontología que formaliza el vocabulario, guían la búsqueda en los repositorios de modelos de seguridad y control de acceso apropiados.
3. Un modelo PIM (*Platform Independent Model*) funcional es derivado del modelo CIM o, alternativamente, ofrecido como entrada funcional a nivel PIM (PIM 1 en la Figura 1). Este modelo describe como son implementados los diferentes servicios.
4. Se utilizan modelos de composición para indicar qué propiedades NF deben asociarse a qué elementos del modelo funcional. Idealmente, una herramienta que automatize el proceso de desarrollo propondría modelos de composición pre-generados.
5. En esta etapa, el modelo iMM (*Integrated Meta Model*, PIM 2 en la Figura 1) independiente de la plataforma, contiene ya toda la información, tanto funcional como NF, sobre la cual realizar diferentes tipos de análisis.
6. A continuación, el modelo iMM es refinado en un modelo PSM (*Platform Specific Model*) específico de plataforma. Esta transformación puede basarse en valores por defecto, a menos que se ofrezca información opcional sobre la plataforma objetivo (indicando preferencias y/o disponibilidad de diferentes características).

7. Finalmente, los descriptores de servicio y políticas de seguridad son derivados del modelo PIM 2. Simultáneamente, se generan el código y la configuración NF.

La idea detras de esta aproximación es que, dentro del modelo de proceso, el diseñador pueda expresar intenciones (primitivas) de seguridad de alto nivel (por ej., no repudio, confidencialidad, o integridad), en lugar de los mecanismos técnicos necesarios para implementarlas (ej., encriptación, firma digital, o *timestamp*). El conocimiento técnico de cuáles son los mecanismos necesarios para conseguir estas intenciones está contenido en las reglas de transformación. De esta forma, se libera al desarrollador funcional de alto nivel de la necesidad de conocimientos técnicos de seguridad.

Las entradas NF están conformadas por distintos modelos de seguridad y control de acceso, previamente desarrollados por expertos, los cuales se encuentran contenidos en repositorios. Estos diferentes modelos NF, apropiados para diferentes contextos y dominios, representan un conjunto de opciones de las cuales el desarrollador puede elegir los más apropiados.

El enfoque propone el desarrollo de cada característica de forma individual. Para esto, se utilizan diferentes lenguajes específicos de dominio, cada uno apropiado a la característica individual en estudio, para definir los aspectos NF. Se piensa en diseños NF genéricos, los cuales representan cómo las diferentes intenciones NF pueden ser realizadas, de manera independiente del sistema actual bajo desarrollo. Estos modelos de características NF serían definidos, validados, y representados de una forma adecuada en la(s) plataforma(s) objetivo por expertos en el campo.

Los expertos deben también asegurarse de que los metadatos adecuados sean asociados a estos modelos NF, y así ser capaces de soportar la búsqueda automática. De esta forma, los modelos NF disponibles en los repositorios, y que sean apropiados a las intenciones especificadas para el sistema, pueden ser descubiertos automáticamente, y ser ofrecidos al modelador.

Los modelos de diseño funcional y NF son integrados en un modelo completo del sistema a través de un proceso de composición de modelos. Los modelos de composición se utilizan para vincular los diseños NF obtenidos de los repositorios en el paso 2 del proceso, con los elementos del modelo funcional sobre los cuales se han establecido intenciones NF. A partir de éstos, se realiza una transformación que, considerando los tres modelos (funcional, NF, y de composición), construye el modelo iMM. El modelo resultante representa un modelo completo del sistema, el cual considera tanto características funcionales como de seguridad y control de acceso, evitando la necesidad de mantener una consistencia entre múltiples modelos.

Diferentes tipos de análisis y métricas, los cuales pueden ser calculados sobre este modelo integrado del sistema, refuerzan la confianza de los desarrolladores en la validez del diseño. Los errores que puedan ser descubiertos en esta etapa serán más económicos de corregir, y el arquitecto del sistema tendrá la oportunidad de consultar a un experto de dominio para corregir cualquier asunto complicado.

Los artefactos de implementación en la plataforma objetivo correspondientes a los aspectos de seguridad son identificados individualmente desde el modelo iMM, de manera independiente de los servicios particulares que se estén diseñando. De esta forma, el sistema en desarrollo ofrece los valores particulares para los distintos elementos de la implementación final, pero los modelos de características NF, y sus representaciones en las distintas plataformas, pueden evolucionar independientemente.

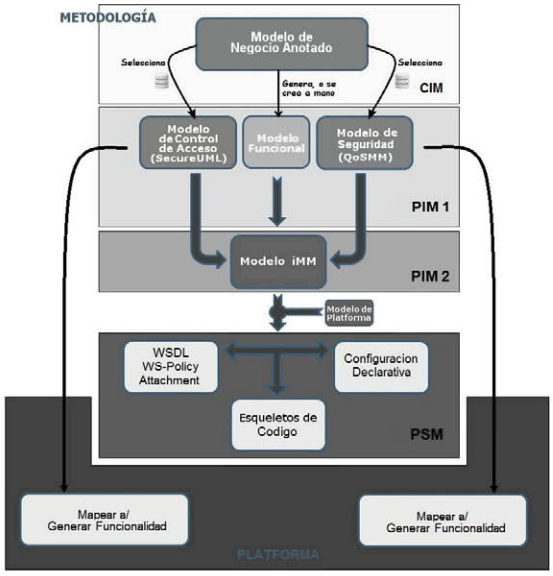

Figura 1: Estructura de la solución propueta.

El beneficio que brinda este enfoque es doble: libera a los desarrolladores funcionales de (la mayor parte de) la necesidad de un conocimiento técnico de seguridad, y permite una mayor reutilización de los modelos de diseño (tanto funcional como NF). Los modelos NF pueden ser reutilizados entre diferentes proyectos, mientras que diferentes configuraciones NF pueden ser aplicadas al mismo modelo funcional, para así obtener diferentes sistemas resultantes.

Un Marco de Desarrollo para la Incorporación de Características de Seguridad

La Figura 1 muestra la estructura general de la aproximación propuesta. Consiste en una cadena de transformaciones y composiciones de modelos (representadas en la figura como flechas). Cada caja en la figura representa un tipo de modelo diferente, entre los cuales se incluyen:

- Un modelo de proceso BPMN, que indica las intenciones de seguridad a alto nivel (CIM).
- Un modelo funcional del sistema (Nivel PIM 1).
- Modelos NF de seguridad y CA (Nivel PIM 1).
- Modelo iMM integrado del sistema (Nivel PIM 2).
- Modelos WSDL, WS-Policy, y XML (Nivel PSM).

Los distintos modelos se describen con más detalle en la sección 3. La idea del marco de desarrollo es la de facilitar un

rápido prototipado. A través del uso de opciones por defecto, un analista del sistema de servicio puede generar un diseño inicial completo del sistema para análisis, prueba, o simulación, y así obtener una rápida realimentación de la validez del diseño.

El metamodelo iMM fue definido con el objetivo de ser capaz de contener, en un único modelo, toda la información necesaria para el análisis y la generación de los diferentes artefactos de salida. Los artefactos resultantes (a los diferentes niveles, incluyendo el modelo iMM) son generados automáticamente. De cualquier forma, el desarrollador tiene la opción de modificar los resultados intermedios para perfilar la generación de los resultados posteriores en la cadena.

3. Una implementación para Dispositivos de la WoT

En esta sección se identifican un conjunto de lenguajes de modelado para tratar aspectos de control de acceso y seguridad. En este artículo solo se presentarán las características de seguridad. El desarrollo del control de acceso es bastante similar. En lo referido a los lenguajes y extensiones empleados e implementados, se han adoptado algunos previamente definidos (SecureUML, QoS, WSDL), otros ya existentes se han integrado para definir unos nuevos, mientras que los restantes han sido definidos desde cero para este caso. Esta sección presenta una implementación del proceso para dispositivos DPWS, y describe como se modela en este caso la seguridad en los niveles CIM, PIM (PIM 1 y PIM 2) y PSM.

Caso de Uso: Control Distribuido de un Aerogenerador. Para ilustrar el uso de la aproximación propuesta, se presenta un ejemplo simplificado de control distribuido de la velocidad de un aerogenerador. En este ejemplo, existen tres participantes (un cliente, la red eléctrica, y dos servicios, el sensor de velocidad del aire, y el servicio de velocidad del rotor del aerogenerador). La red eléctrica determina, en base a la necesidad de energía y a la velocidad del viento (servicio *windSpeedService*), cual es la velocidad apropiada para el rotor del aerogenerador (servicio *setSpeedService*). La Figura 2 presenta el modelo de procesos que describe estos servicios.

3.1. Modelado a Nivel CIM

Modelo de Proceso del Aerogenerador

Un diagrama BPMN que modela el proceso presentado en la Sección 3 se incluye en la Figura 2. Nos centraremos el el servicio *WindSpeedService*. Tras una evaluación del proceso, se determina que es importante asegurar a un grado medio la integridad de los mensajes *velocidadVientoResp* (que participa en la determinación de la velocidad a la cual fijar el rotor), y se establece que también es de interés brindar confidencialidad al servicio. Finalmente, se asocia al servicio la característica de seguridad de auditoría (*Auditing*), para mantener un registro de actividades del dispositivo.

Las características de seguridad requeridas son asociadas al envío de estos dos mensajes mediante notas de texto. Se estudia el contenido de notas de texto presentes en el modelo para determinar si son anotaciones de seguridad, o simples notas. En

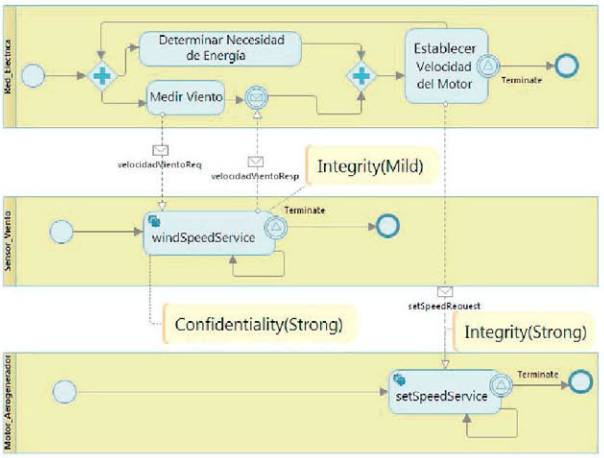

Figura 2: *Modelo de Proceso de Control de Aerogenerador*

el caso de modelos de actividad UML, un perfil sería el mecanismo más apropiado a utilizar.

Modelos de Proceso. Los diagramas de procesos se han convertido en la alternativa dominante a la hora de modelar composiciones e interacciones de servicios, como se indica en la Sección 4.1.

El modelo de procesos se corresponde con el punto 1 del proceso presentado en la Sección 2. El objetivo de la propuesta es ofrecer medios para indicar los requisitos de seguridad de una composición de servicios, para más tarde guiar la implementación de mecanismos que cumplan dichos requisitos. Los elementos que están sujetos a requisitos de seguridad incluyen a los que representan servicios, operaciones, y los mensajes entre éstas. Adicionalmente, para facilitar la expresión de las intenciones de seguridad comunes a varios de estos elementos, también se soporta la anotación de elementos de agrupación, considerando a las intenciones de seguridad aplicadas a todos los elementos contenidos dentro del grupo.

En el caso de los modelos de proceso BPMN como el utilizado en este ejemplo, los elementos a anotar incluyen:

- Eventos y Flujos de Mensaje, Tareas de Servicio, Objetos de Datos.
- Agrupaciones: Grupos, Piscinas, Carriles.

Vocabulario de Intenciones de Seguridad

En esta estapa, Nos encontramos en el punto 2 del proceso, a nivel CIM. A este nivel, las intenciones abstractas de seguridad deberían permitir a los diseñadores del sistema expresar los requisitos de seguridad sin necesidad de un conocimiento técnico detallado de los mecanismos de implementación.

Las ontologías se han convertido en herramientas muy útiles para acordar y adoptar un lenguaje común para describir y descubrir elementos (servicios web, por ejemplo). El uso de una ontología en el nivel CIM puede ofrecer un mecanismo no solo

para definir un vocabulario común para las intenciones de seguridad, sino para el proceso de identificación de los elementos que cumplen los requisitos impuestos, y su grado de cumplimiento (*matchmaking*), de los modelos NF contenidos en los repositorios que puedan cumplir los requisitos NF impuestos.

En el dominio de la seguridad, la ontología NRL [Kim et al. (2007)] representa una opción muy completa, que parte desde objetivos de seguridad abstractos de alto nivel, a través de credenciales de seguridad, hasta llegar a algoritmos técnicos de encriptado. Esta ontología ha sido elegida para formalizar el vocabulario de seguridad en esta implementación.

Tras un estudio exhaustivo de estándares pertinentes (ej., ISO/IEC 25010 [ISO/IEC (2011)]), y trabajos de investigación sobre el modelado de primitivas y requisitos de seguridad, se ha definido un lenguaje de intenciones abstractas de seguridad (en inglés), el cual se presenta en la Tabla 1. El objetivo de este vocabulario es el de permitir ejemplificar el uso del proceso.

En la Tabla 1, la primer columna presenta las cinco intenciones de seguridad definidas: Autorización, Confidencialidad, Integridad, Auditoría, y No Repudio. La segunda columna presenta los diferentes atributos definidos para cada una de las intenciones, los cuales brindan una mayor granularidad. La Tabla 1 también presenta, en su tercer columna, las correspondencias de estas intenciones en los objetivos de seguridad definidos por la ontología NRL.

Los atributos incluidos en el vocabulario permiten refinar las intenciones de seguridad impuestas sobre los elementos. De esta forma, puede especificarse un cierto grado de confidencialidad de mensaje, ya sea fuerte (*Strong*), medio (*Mild*), o débil (*Soft*). Para ser capaces de representar estos atributos en una implementación en particular, es necesaria una categorización de los distintos algoritmos de criptografía disponibles. La categorización adoptada en este trabajo es la que se presenta en [Asnar et al. (2009)].

3.2. Modelado a Nivel PIM 1

Metamodelo de Seguridad: QoS Metamodel

El perfil QoS [OMG (2008)] de UML define el metamodelo *QoS Framework Metamodel* como una extensión del de UML. Sin embargo, los conceptos allí definidos pueden ser utilizados para modelar características NF como un lenguaje específico de dominio. Este ha sido el metamodelo elegido para representar los elementos de seguridad en el enfoque que se presenta aquí.

Modelo QoS de Seguridad DPWS. Idealmente, en un entorno de desarrollo ya instalado, los respositorios contarían con un número de modelos adaptados a las diferentes características NF de los diferentes dominios, de entre los cuales poder elegir y reutilizar los más acordes. En este caso, dicho entorno de desarrollo no se encontraba presente, haciendo necesario el desarrollo de un modelo QoS específico para representar las características de seguridad definidas para el perfil DPWS.

Búsqueda de Modelos de Seguridad

En este punto, la herramienta debe determinar si existen en los repositorios que aparecen en el nivel CIM de la Figura 1, modelos de seguridad que satisfagan las intenciones de seguridad expresadas en el modelo de proceso. La base para poder realizar esta búsqueda automáticamente es haber asociado a estos modelos los metadatos apropiados. En este ejemplo, los modelos de seguridad han sido anotados a nivel de modelo con los distintos objetivos de seguridad (definidos por [Kim et al. (2007)]) a los cuales se enfocan. A nivel de elemento se han anotado con las clases correspondientes, definidas en la ontología, que soportan esos objetivos. Como mecanismo de anotación se han utilizado los perfiles EMF (EMF Profiles, Langer et al. (2011)), los cuales permiten extender los modelos EMF con información que no ha sido prevista en el metamodelo (en este caso, se extienden los modelos con información de la ontología NRL). Se asume, para continuar con el ejemplo, que existe en el repositorio un modelo de seguridad QoSMM adaptado al perfil DPWS, que ofrece mecanismos que satisfacen los requisitos de Integridad.

La Tabla 2 presenta una parte de las anotaciones semánticas (diferentes clases en la ontología) apropiadas para la intención de integridad, de acuerdo con el perfil DPWS. Se incluyen en la tabla, en este caso, únicamente las clases especificadas como requeridas para la seguridad en el perfil DPWS.

En lo referido a la seguridad, el perfil DPWS dicta unos requisitos mínimos, pero no impide que otras características mejores sean utilizadas. Por ejemplo, se especifica claramente en el estándar, en la Sección 6.6.1 (requisitos R4059-R4062, y R5014 de [OASIS (2009)], un conjunto de algoritmos de cifrado que **deben** soportarse, otros que **se recomienda** sean soportados, y se especifica que otros restantes **pueden** ser soportados (y se recomienda soportar algunos más robustos), mientras que se indican algunos que **no deben** negociarse/utilizarse para la seguridad. Estas recomendaciones son incluidas en el desarrollo, y se corresponden con las opciones por defecto de la generación. Como se mencionó anteriormente, las diferentes intenciones de seguridad de nivel CIM se corresponden con estos objetivos de seguridad como se indica en la Tabla 1.

Los repositorios podrían contener otros modelos de seguridad, apropiados a otros dominios, y probablemente modelados en otros lenguajes, que pudieran enfocarse en satisfacer los mismos objetivos de seguridad. El mecanismo de búsqueda los presentaría al desarrollador como posibles opciones, y sería tarea de éste el elegir el más apropiado.

Modelo de Entrada Funcional: UML

UML, el estándar de modelado de facto, es soportado como entrada funcional a nivel PIM si se prefiere presentar un diseño. Estando enfocada la solución hacia los servicios web, el perfil SoaML [OMG (2009)] para UML ha sido implementado, y es aplicado a estos modelos de entrada para guiar la transformación UML a iMM. Esto es compatible con el uso de otros perfiles UML, por ejemplo, el perfil MARTE [OMG (2007)], que pueda guiar la generación del código embebido.

Un modelo funcional a este nivel corresponde al punto 3 del proceso. En el ejemplo empleado en este artículo, se asume que el desarrollador no cuenta con un modelo funcional UML de implementación. Se deriva un modelo iMM funcional a partir del modelo de proceso, y se procede con el resto de los pasos

Tabla 1: Vocabulario de Intenciones de Seguridad

Intención	Atributos	Ontología NRL
Authorization	Par{Atributo, Valor}	Authorization
Confidentiality	Nivel (Strong, Mild, Soft)	Confidentiality
Integrity	Nivel (Strong, Mild, Soft)	MessageIntegrity
Auditing		UserAuthentication, MessageAuthentication
Non-Repudiation	Sin Atributos	UserAuthentication, MessageIntegrity, MessageAuthentication, ReplayPrevention

Tabla 2: *Clases Semánticas que Soportan Integridad y Firma (parcial)*

Mecanismo	Anotación (Ont. NRL)	Objetivo de Seg. NRL
SSL	SecurityProtocol/ NetworkSecurityProtocol/RSA	Confidentiality UserAuthentication
Algoritmo RSA	EncryptionAlgorithm/ AsymmetricAlgorithm/RSA	Confidentiality
Algoritmo RC4	EncryptionAlgorithm/ SymmetricAlgorithm/RSA	Confidentiality
Algoritmo SHA-1	SignatureAlgorithm/ HashAlgorithm/SHA-1	MessageIntegrity MessageAuthentication
Credencial X.509	Credential/ElectronicToken/ Certificate/X.509Certificate	UserAuthentication

a partir de éste. El modelo iMM funcional derivado define los servicios, sus interfaces, operaciones, los mensajes, parámetros, y componentes de implementación, a partir de los elementos en el modelo BPMN. Las correspondencias son:

- una *piscina* BPMN → un sistema,
- un *carril* BPMN → un componente de implementación de servicio ,
- una *actividad de servicio* BPMN → un servicio,
- operaciones → operaciones,
- mensajes → mensajes.

Composición de los Modelos Funcional y de Seguridad

En esta etapa, correspondiente al punto 4 del proceso, ya contamos con un modelo funcional (provisto por el desarrollador, o un modelo tentativo derivado del modelo CIM) y un modelo de seguridad que ofrece mecanismos para satisfacer las intenciones de seguridad inicialmente especificadas. En base a las anotaciones semánticas de los elementos del modelo QoSMM (indicadas en la segunda columna de la Tabla 2), pueden proponerse modelos de composición tentativos (generados automáticamente por la herramienta), asociando los mecanismos que satisfacen las intenciones de seguridad, con los elementos funcionales correspondientes.

Lenguajes de Composición. Los lenguajes de composición definen los mecanismos de asociación que pueden utilizarse entre los diferentes modelos. En este caso, la composición de modelos funcionales con los modelos de control de acceso (CA) o los de seguridad requieren asociaciones de composición diferentes, como se describe en [Silva Gallino et al. (2010)]. La semántica de las asociaciones de composición se implementa como parte de las reglas de las transformaciones de composición que guían el proceso.

En el caso de las características de seguridad, la composición agrega al elemento anotado (en el ejemplo, el servicio *WindSpeedService*) una referencia a un nuevo elemento de política de seguridad que será generado en el modelo iMM. Este nuevo elemento contiene los datos que definen la característica de seguridad, los cuales son utilizados para la generación de la implementación final.

Para el control de acceso (no presentado en este artículo), el modelo de composición asocia el elemento anotado a *acciones* (por ejemplo, ejecutar el servicio) descritas en el modelo de control de acceso. Como consecuencia, a partir de la composición, se generan en el modelo iMM nuevas acciones correspondientes a este recurso, y se asocian a éste los distintos permisos de acceso.

Los elementos que pueden tener características de seguridad asociadas son los correspondientes a los elementos anotables en el modelo de proceso: un sistema (una *piscina* BPMN), un componente de implementación (un *carril* BPMN), un servicio (una *actividad de servicio* BPMN), operaciones, y mensajes.

Composición del Servicio de Aerogenerador

Para lograr proponer modelos de composición tentativos son necesarios tres elementos: la identificación de correspondencias entre el vocabulario de intenciones de seguridad y la ontología NRL (Tabla 1), el algoritmo de búsqueda propuesto por la propia ontología NRL [Kim et al. (2007)], y una categorización de la fortaleza de los algoritmos [Asnar et al. (2009)].

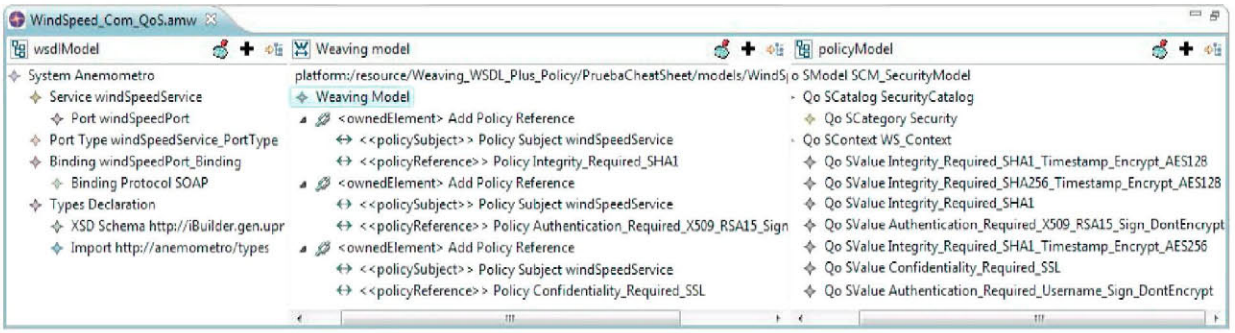

Figura 3: *Modelo de Composición Funcional y de Seguridad*

Por ejemplo, un elemento del modelo QoSMM que especifique el uso del algoritmo SHA-1, soporta (como se indica en la Tabla 2) los objetivos de seguridad de *MessageIntegrity* y *MessageAuthentication*. De igual forma, y siguiendo la categorización de [Asnar et al. (2009)], el algoritmo SHA-1 se correspondería con el atributo *Mild* de la intención de *Integridad*, satisfaciendo así las intenciones expresadas sobre los servicios en el diagrama de la Figura 2.

Puede observarse, en la Figura 3, como se realiza la composición entre el modelo iMM funcional derivado (panel izquierdo), el modelo QoS de seguridad (panel derecho), y el modelo de composición que los vincula (panel central). Puede observarse en la columna central, como se asocia al elemento *WindSpeedService*, las características de seguridad de *Integridad SHA-1* (correspondiente a la anotación de Integridad de la Figura 2), *Confidencialidad SSL* (correspondiente a la anotación de ConfidencialidadFuerte), y *Autenticación X.509 RSA* (necesaria para SSL, y la del algoritmo de mayor robustez entre los disponibles, RSA y RC4, dado el atributo). La interfaz de composición que se observa en la Figura 3 es la que presenta la herramienta AMW [Didonet del Fabro et al. (2005)], una herramienta específica de composición de modelos.

Los modelos tentativos de composición generados automáticamente podrían, como todo modelo que participa del proceso, ser modificados/adaptados a las necesidades del sistema puntual que se está desarrollando. De esta forma, los desarrolladores pueden ajustar e influenciar la posterior generación de artefactos resultantes (configuración, código) para lograr un resultado más adaptado a sus necesidades.

En la composición del modelo funcional con el modelo de seguridad pueden incluirse multiples alternativas de seguridad, independientemente de si la plataforma objetivo final las soporta o no. El mecanismo para filtrar las alternativas no válidas a la hora de la transformación a la implementación final es el uso del modelo de plataforma WS-SecurityPolicy. Las alternativas modeladas a nivel PIM, y que no aparezcan en este modelo de plataforma, no serían incluidas en la implementación final. En caso de que ninguna alternativa sea soportada, una alerta debería notificar al desarrollador de que la implementación no cumple con el modelo de diseño. De esta forma se facilita la posibilidad de modelar un único diseño del servicio, que luego puede implementarse sobre distintas plataformas.

3.3. Modelado a Nivel PIM 2

Lenguaje iMM

El metamodelo intermedio iMM es una de las piezas fundamentales del sistema, y se corresponde con el punto 5 del proceso. Esta implementación en particular, descrita en [Silva Gallino et al. (2010)], ha sido diseñada para ser lo más general posible. Se ha buscado maximizar el soporte de estándares, técnicas, y lenguajes de CA y políticas de servicio. Está compuesto por tres partes diferenciadas: Diseño funcional, Control de Acceso, y Políticas de Servicio.

La parte funcional del metamodelo sigue un desarrollo basado en componentes, y es utilizado para describir componentes e interfaces de servicio, entre otras metaclases. Se ha incluido un metamodelo genérico de CA [Mouelhi et al. (2010)] que soporta un gran número de técnicas de CA, y transformaciones entre estas. Finalmente, se ha elegido parte del paquete "Policy" del metamodelo *CBDI-SAE Meta Model for SOA* [Dodd et al. (2007)] para representar las políticas de servicio. Los tres metamodelos elegidos fueron estudiados en detalle para determinar conceptos equivalentes y otros puntos de unión entre ellos.

Un metamodelo integrado, como lo es iMM, representa un excelente sujeto para el análisis y cálculo de métricas. Para iMM en particular, se han definido un análisis de conflicto de autorización (cuando se modela control de acceso), métricas de evaluación de cobertura de seguridad de las interfaces de servicio, o de evaluación de satisfacción de los requisitos de nivel CIM (trazabilidad CIM-PIM). Los análisis y métricas requieren una extensa descripción y análisis, y por ello no serán presentados en este artículo. El lector puede referirse a [de Miguel et al. (2008)], donde se presenta un estudio de cómo se pueden realizar este tipo de análisis y cálculo de métricas en modelos.

Modelo iMM del Aerogenerador

Una vez realizada la composición de modelos, una transformación modelo a modelo (M2M) genera el modelo integrado iMM del sistema. Esta transformación toma como entradas el modelo funcional del sistema, el modelo de seguridad QoSMM, y el modelo de composición que los vincula. Estos tres modelos son los que aparecen en la Figura 3. Las reglas de transformación definidas para los lenguajes empleados para definir los modelos (en este caso, reglas de transformación de QoSMM a iMM, por ejemplo) guían dicha transformación. El conjunto de

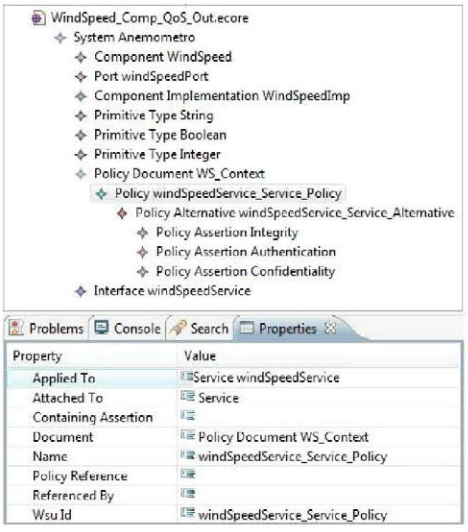

Figura 4: *Modelo iMM del Sistema Anemómetro*

reglas correspondiente a cada lenguaje solo necesitan definirse una única vez, y posteriormente pueden ser utilizadas para todo modelo (funcional o NF) definido con este lenguaje.

El modelo iMM representa, entonces, un diseño completo (considerando tanto aspectos funcionales como NF). La Figura 4 presenta el modelo iMM para el dispositivo *Anemómetro*.

3.4. Generación de los Artefactos Resultantes

Los artefactos resultantes del proceso de desarrollo incluyen descriptores WSDL y WS-Policy, y la implementación en el lenguaje soportado por el marco de trabajo (C, C++, Java), incluida la seguridad correspondiente. La generación de estos elementos se corresponde con los puntos 6 y 7 del proceso. Opcionalmente se puede contar con un modelo WS-SecurityPolicy de plataforma de seguridad, que indica la disponibilidad de algoritmos o credenciales de seguridad soportados, limitando las reglas de transformación aplicables (definidas en base al perfil DPWS) a un subconjunto acorde a los elementos disponibles.

Lenguaje WSDL. Se ha utilizado el lenguaje WSDL para modelar elementos en las transformaciones y composiciones. Su uso facilita la visualización de las interfaces externas de servicio ofrecidas por el dispositivo, y la especificación de puntos de aplicación de las políticas de servicio.

Lenguaje WS-Policy. De manera equivalente, se ha desarrollado un lenguaje WS-Policy. Las distintas transformaciones y posteriores generaciones de archivos de configuración NF operan sobre este metamodelo, incluyendo el caso de los modelos WS-SecurityPolicy descritos a continuación.

Lenguaje WS-SecurityPolicy. Se ha definido un lenguaje WS-SecurityPolicy como extensión del modelo WS-Policy anteriormente mencionado, para ofrecer un mecanismo práctico para la edición y validación de configuraciones de seguridad, y como medio para indicar la disponibilidad de características, capacidades, y preferencias de seguridad de las plataformas objetivo.

Platforma Objetivo: Perfil de Dispositivos para Servicios Web

El perfil de servicios web para dispositivos (DPWS) [OASIS (2009)] representa un subconjunto de estándares enfocados en establecer unas funcionalidades mínimas necesarias que permitan lograr interoperabilidad entre servicios ofrecidos y/o consumidos por dispositivos con recursos de hardware limitados, y otros clientes con mayores funcionalidades. Este perfil busca limitar los protocolos y formatos, haciendo posible la implementación de los WS en dispositivos electrónicos de consumo, por ejemplo, estableciendo unos requisitos mínimos, pero sin limitar la posibilidad de ofrecer implementaciones más ricas en funcionalidad.

En particular, busca ofrecer cuatro funcionalidades básicas:

- Intercambio de mensajes seguros entre servicios web.
- Descubrimiento dinámico de los servicios.
- Descripción de dichos servicios.
- Suscripción a eventos lanzados por estos servicios.

En lo referido a la seguridad, el perfil no dicta una política de seguridad estricta, sino que especifica una serie de pautas, como ser el uso de:

- Canales seguros punto a punto TLS/SSL.
- Certificados X509 para identificación de dispositivos.
- Firma digital para la integridad de los mensajes.

Implementación DPWS. Actualmente existen diferentes marcos de desarrollo disponibles que implementan el perfil DPWS:

- Microsoft WSDAPI [Microsoft (2012b)] y .NET Micro Framework DPWS implementation [Microsoft (2012a)].
- Web Services for Devices (WS4D) [WS4D (2007)] project: C/C++, JavaMEDS, Java/Axis2, Java/CoAP.
- SOA4D [SOA4D (2007)](Service-Oriented Architecture for Devices).

Los marcos de trabajo ofrecen mucha funcionalidad que, de otra manera, debería ser generada para el servicio. Esto simplifica sustancialmente el desarrollo de los generadores de código. Para este ejemplo, se ha seleccionado el marco *WS4D JavaMEDS* para soportar la implementación.

WS4D JavaMEDS divide la implementación en tres elementos diferentes: dispositivos, servicios, y operaciones. Un dispositivo ofrece un número de servicios, que a su vez presentan una cantidad diferente de operaciones. La Figura 5 muestra la implementación sobre este marco del dispositivo *Anemómetro*, la cual nos permite visualizar mejor las diferencias de implementación cuando se ha asociado o no seguridad a un dispositivo.

El código que no ha sido recuadrado en la figura corresponde al modelo funcional. Éste está presente en tanto si la implementación incluye características de seguridad, como si no. Los

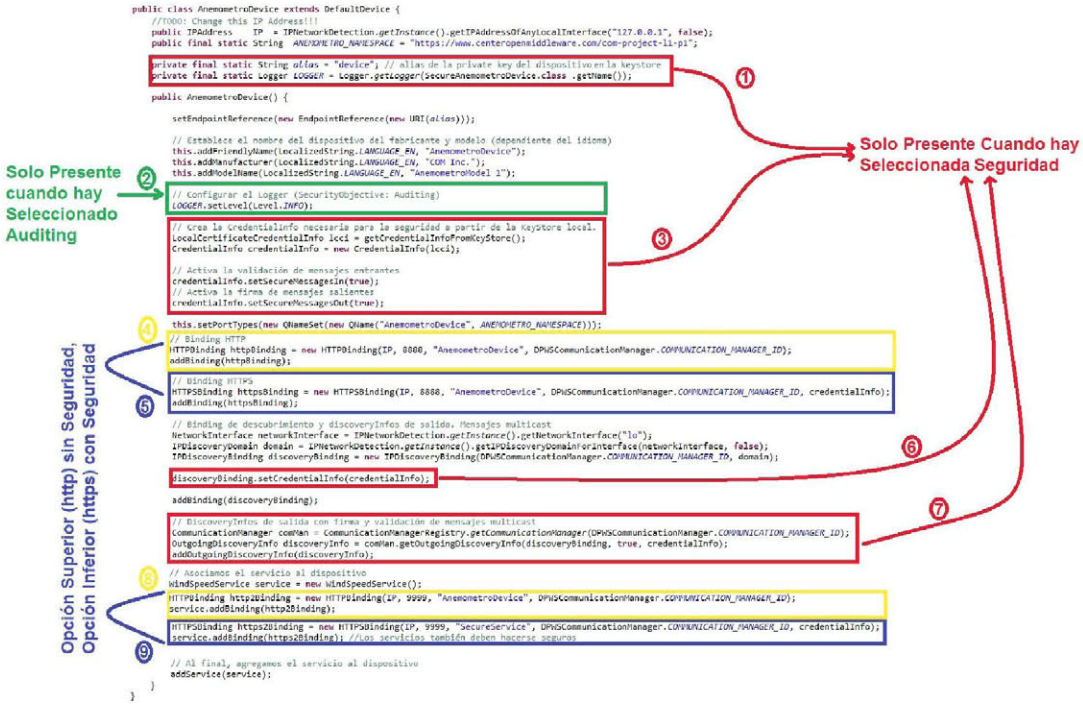

Figura 5: *Código de Implementación JavaMEDS*

recuadros numerados 1,3,6, y 7 corresponden a lineas de código que solo aparecen si se asocian características de seguridad al dispositivo. De esta forma, el código del recuadro 1 define constantes para describir el alias del dispositivo en el respositorio de claves (*Keystore*), o el mecanismo de registro de eventos, si se ha solicitado la característica *Auditing*.

Si se ha asociado la característica *Auditing* al servicio, es necesario configurar un registro de eventos. Esta configuración se realiza, por ejemplo, mediante el código contenido dentro del recuadro 2. El código dentro del recuadro número 3 se encarga, primero, de obtener la información de la credencial del dispositivo a partir del repositorio de claves local, y, a continuación, verificar (a partir de la característica *MessageIntegrity*) la existencia de una firma digital en los mensajes entrantes y salientes. De la misma manera, el código contenido en los recuadros 6 y 7 asocia la información de la firma digital a los intercambios de mensajes de descubrimiento del dispositivo.

Los recuadros 4 y 5, y 8 y 9 respectivamente, son un caso diferente. En esta ocasión, el código a incluir es diferente en caso de que se haya solicitado el uso de TLS/SSL para asegurar la conexión punto a punto. De este forma, si la característica *Confidentiality* ha sido asociada al servicio, el código a incluir en el dispositivo corresponde al contenido en los recuadros número 5 y 9, mientras que si esta característica de seguridad no le ha sido asociada, el código a incluir es el que se encuentra dentro de los recuadros 4 y 8.

Lo anteriormente presentado representa la implementación en código de las características modeladas. Los elementos correspondientes a los descriptores WSDL y WS-Policy del ejemplo presentado no serán incluidos en este artículo. El lector puede referirse a [Silva Gallino et al. (2011b)] para una presentación de la generación de los distintos artefactos de configuración para un caso de uso de servicos de cadena de suministros.

El ejemplo presentado, aunque muy brevemente descrito, brinda una idea de los mecanismos que ofrece esta aproximación para satisfacer los objetivos fijados en la sección 1. Los distintos campos y elementos de los artefactos resultantes representan las transformaciones de los distintos DSLs a lo largo del proceso. Por otra parte, es el modelo iMM del sistema el que fija los valores puntuales de estos elementos resultantes, determinando la configuración e implementación específica de cada dispositivo. De esta forma, diferentes modelos funcionales y NF, independientemente gestionados (mantenidos, evolucionados, representados en plataformas objetivo) pueden reutilizarse y combinarse de formas distintas, para generar implementaciones particulares de los sistemas.

4. Trabajo Relacionado

4.1. Requisitos No-Funcionales

Existen varios trabajos que estudian como especificar requisitos de seguridad en modelos de proceso [Menzel y Meinel (2009); Satoh et al. (2008); Wada et al. (2008)]. Algunos de ellos han sido considerados para definir la lista de intenciones abstractas de seguridad que se utilizan en este enfoque. Estos trabajos comparten la característica de que se utilizan perfiles

o metamodelos específicos individuales para modelar las propiedades de seguridad del sistema. En nuestro trabajo, sin embargo, se utilizan múltiples metamodelos independientes para describir las características, proponiendo el desarrollo de cada característica de forma individual, de una manera similar a lo propuesto en Multi-Dimensional Separation of Concerns [Tarr et al. (1999)], facilitando la comprensibilidad, análisis, y generación de los diferentes aspectos individuales del servicio.

Existen otras aproximaciones que buscan representar los requisitos NF de los servicios a alto nivel, tales como el marco de trabajo NFR (*Non-Functional Requirements*) *Framework* [Chung et al. (2000)]. En éste, los requisitos NF son representados como objetivos (*soft-goals*), y se define el concepto de *satisficing* (contribuciones parciales a favor o en contra de satisfacer el objetivo). El marco de trabajo NFR representa una alternativa a los modelos de proceso a alto nivel de abstracción.

4.2. Modelado de Sistemas DPWS

[Theorin et al. (2012)] propone el uso de *Grafchart* (un lenguaje gráfico enfocado a aplicaciones de control secuencial) para la definición de procesos de automización de control basados en servicios. *Grafchart* es un lenguaje de flujo, enfocado en la definición de la secuencia funcional del proceso. La aproximación aquí propuesta intenta brindar una solución que cubra tanto los aspectos funcionales como NF de los servicios.

[Illner et al. (2005, 2006)] presentan MoBaSeC, una herramienta para la gestión, guiada por modelos, de la seguridad de sistemas de dispositivos de software empotrado basados en servicios, principalmente orientados al control de acceso. El enfoque está estructurado en tres niveles: roles y objetivos, servicios y configuración, y dispositivos y archivos. La solución de Illner et al. (2005) se orienta fundamentalmente al control de acceso. La solución propuesta en las secciones anteriores busca soluciones integrables con otros tipos de característica NF.

4.3. Propiedades No-Funcionales en Servicios Web

Existen múltiples trabajos enfocados al desarrollo guiado por modelos de características NF de los servicios web en áreas no exclusivas a los dispositivos DPWS. De entre las propuestas que trabajan las distintas características NF soportadas por WS-Policy, el enfoque de [Ortiz y Hernández (2006)] es uno de los más representativos. Su aproximación hace uso de la orientación a aspectos, modelado *Service Component Architecture* (SCA) y un perfil UML, para modelar los servicios desde un nivel PIM a modelos específicos. A diferencia del trabajo de Ortiz, en el nuestro se hace uso de lenguajes específicos de dominio (DSLs), más adaptados a cada uno de los aspectos NF a ser tratados, para posteriormente combinar estos modelos DSL en un único modelo integrado.

4.4. Modelado de la Seguridad

En [Meiko Jensen and Sven Feja (2009)], se sugiere la utilización de una *vista de seguridad* (una vista del sistema donde se asocian propiedades de seguridad a elementos funcionales del sistema) en la cual se modela la seguridad a un nivel técnico. Nuestro trabajo comparte la idea de [Meiko Jensen and Sven Feja (2009)] de utilizar distintas vistas o modelos para representar las características NF de los sistemas, pero se diferencia en (entre otras cosas) que el último no hace uso de las cadenas de transformación abstracto a específico que propone MDA, no beneficiándose de las posibilidades de reutilización de diseño que este paradigma ofrece. Por otra parte, nuestro trabajo permite también aislar al desarrollador funcional de la necesidad de conocimientos técnicos de seguridad.

[Satoh et al. (2008)] presenta una metodología para lograr una configuración de seguridad extremo a extremo en sistemas basados en servicios. Se propone una aproximación guiada por modelos, y plantillas para especificar patrones de seguridad. Los requisitos de seguridad del sistema son incluidos en modelos de proceso mediante palabras claves abstractas. Estas intenciones de seguridad son complementadas mediante un modelo de infraestructura de seguridad (*Security Infrastructure Model*, SIM), para luego generar una configuración concreta. El modelo SIM, aunque más cercano a un modelo de topología, cumple una función equivalente a la del modelo WS-SecurityPolicy de plataforma aquí propuesto.

Este enfoque, y el presentado en [Satoh et al. (2008)], difieren en la que posiblemente sea la mayor fortaleza de la aproximación aquí presentada: el uso de modelos independientes de diseño de características funcionales y NF, lo que permite la reutilización y favorece la modularidad. En nuestro trabajo, el uso de lenguajes específicos de dominio, cada uno ajustado a la característica NF que describe, facilita la comprensión, análisis, y generación individual de los distintos aspectos del servicio.

En lo referido a la ontología de seguridad utilizada para la anotación de los modelos de los repositorios, en nuestro trabajo se ha utilizado la ontología NRL [Kim et al. (2007)]. Existen ontologías de seguridad más enfocadas a los sistemas de control industrial [Nabil y Mohamed (2012)], cuya utilización aún no ha sido evaluada.

La Tabla 3 presenta una comparativa del trabajo relacionado.

5. Conclusiones y Trabajo Futuro

Este artículo presenta una propuesta de aplicación a dispositivos de software empotrado que cumplen el perfil DPWS, de un proceso de desarrollo guiado por modelos para el soporte de características de seguridad de sistemas orientados a servicios. En este enfoque, las ontologías son empleadas para soportar el descubrimiento automático de modelos de seguridad que satisfagan los objetivos alojados en los repositorios, y para proponer modelos de composición tentativos.

Dentro de esta aproximación, las características de seguridad se abordan a un nivel de abstracción apropiado a cada etapa del desarrollo. Múltiples modelos específicos de dominio, previamente desarrollados por expertos, enfocados, de forma independiente, a distintas características NF, y que pueden ser gestionados y evolucionados también independientemente, facilitan la reutilización de diseño, y favorecen el que cada característica sea abordada de la forma más conveniente. Los arquitectos funcionales son así aislados de los aspectos NF del

Tabla 3: *Comparativa de Trabajo Relacionado*

Referencia	Perfil /DSL	NF	Niveles MDA	Vistas	Modular	PM
[Menzel y Meinel (2009)]	DSL	Seg.	CIM, PIM, PSM	Única	No	No
[Chung et al. (2000)]	DSL	Múltip.	CIM	Única	No	No
[Theorin et al. (2012)]	DSL	No	Único	Única	No	No
[Illner et al. (2005)]	DSL	C.A.	CIM, PIM, PSM	Única	No	No
[Ortiz y Hernández (2006)]	Perfiles	WS-Policy	PIM, PSM	Única	Si	No
[Meiko Jensen and Sven Feja (2009)]	DSL	C.A., Seg., +	Único	Múltip.	Si	No
[Satoh et al. (2008)]	Perfil y DSL	Seg.	CIM, PIM, PSM	Única	No	Si
Este Trabajo	Perfiles y DSL	C.A., Seg., +	CIM, PIM, PSM	Múltip.	Si	Si

sistema, aliviándolos de la necesidad de conocimientos técnicos NF. Son también aislados de las características de, en este caso, seguridad, lo que permite limitar el conocimiento del diseño de seguridad a un número reducido de desarrolladores de confianza. Finalmente, el modelo integrado del dispositivo ofrece un sujeto sobre el cual ejecutar análisis de forma práctica, y en etapas tempranas del desarrollo.

Gracias a su independencia de la plataforma, el uso de un enfoque basado en MDSoC, y políticas para indicar las características NF, el enfoque propuesto presenta flexibilidad en su evolución en lo referido a la aparición de nuevos estándares o tecnologías. Ante esto, solamente sería necesario el desarrollo de nuevos generadores para la plataforma objetivo específica. Es la creencia de los autores que el enfoque aquí presentado puede utilizarse para otras características NF más allá de la seguridad, abriendo la puerta a evaluar su aplicación al desarrollo de DPWS no solo con seguridad, sino que incluya conceptos de confiabilidad o requisitos temporales, por ejemplo.

La seguridad basada en WS-Security [OASIS (2006)] implica una carga importante en lo referido al tamaño de los mensajes y en el esfuerzo de análisis sintáctico de éstos con el objetivo de lograr confidencialidad, integridad, y autenticación [Unger et al. (2012)]. [Shopov et al. (2007)] ofrece una comparación del rendimiento de WS-Security vs TLS/SSL en sistemas de software empotrado. No todos los dispositivos empotrados disponen de las capacidades de memoria, computacionales, o de almacenamiento suficientes para soportar estas cargas [Blet y Simón (2011)].

Sin embargo, ya hoy existen dispositivos empotrados que sí presentan estas capacidades, y es esperable que su número vaya en aumento. Además, comienzan a surgir alternativas, como las propuestas en [Hernandez et al. (2009); Unger et al. (2012)], que mantienen la compatibilidad con WS-Security a la vez que mejoran la eficiencia, algo muy importante para dispositivos de software empotrado, donde los recursos suelen ser limitados. En ambientes donde la seguridad extremo a extremo es importante, el uso de WS-Security es de gran utilidad.

A corto plazo, el esfuerzo se concentrará en finalizar la automatización de las distintas etapas de la cadena CIM-PIM-PSM (búsqueda automática, modelos de composición tentativos), y en la validación de las primitivas abstractas de seguridad. También como parte del trabajo futuro, se piensa aplicar este enfoque a un caso de uso de control industrial.

Por otra parte, se han definido ciertos análisis para modelos iMM, los cuales no han sido incluídos en este artículo. El desarrollo de otro tipo de análisis a este nivel, en conjunto con el uso de transformaciones horizontales del modelo de diseño a otras representaciones capaces de soportar formalización o simulación, figuran como líneas futuras que despiertan gran interés.

English Summary

A Model-Driven Strategy for Including Security Aspects in Web Services-Based Embedded Services

Abstract

In modern distributed systems, such as in the Internet or Web of Things, security plays a fundamental role. Special atention must be placed, then, in considering these aspects in the first stages of development. In this context, the model-driven development of non functional (NF) requirements is of great interest, as it addresses those NF characteristics in the design stage, when analyses can be performed, and there is room for changes while they are still not too costly. The use of model-driven methodologies brings with them some intrinsic benefits, such as the increase in productivity, a greater reuse of design elements, or an improved maintainability of the system. This paper presents a development strategy that allows integrating non-functional security aspects (such as as confidentiality, integrity, or access control) in embedded systems design.

Keywords:
MDD DPWS WS-Policy WS-SecurityPolicy

Agradecimientos

Este trabajo ha sido financiado en parte por el Centro Español de Desarrollo Tecnológico (CDTI, Ministerio de Industria, Comercio y Turismo), por medio del proyecto ITECBAN (Infraestructura Tecnológica y Metodológica de Soporte para un Core Bancario), del programa INGENIO 2010, y por el Ministerio Español de Educación y Ciencia, por medio del proyecto RT-MODEL (Plataformas de tiempo real para diseño de sistemas empotrados basado en modelos, TIN2008-06766-C03-03) del Plan Nacional de I+D+I.

Agradecemos también a la doctora Concepción Sanz Pineda por sus sugerencias y comentarios acerca de este artículo.

Referencias

Asnar, Y., Felici, M., Kokolakis, S., Li, K., Saidane, A., Yautsiukhin, A., 2009. Serenity Project Deliverable A1.D5.1 - Preliminary version of S&D Metrics.

Blet, N. S., Simón, J. L., 2011. SOA en automatización de pymes manufactureras. Iberoamericana de Engenharia Industrial [2175-8018] 3 (2), 190.

CDTI, 2006. ITECBAN, Infraestructura Tecnológica y Metodológica de Soporte para un Core Bancario.
URL: http://www.daedalus.es/i-d-i/proyectos-nacionales/itecban/

Chung, L., Nixon, B. A., Young, E., Mylopoulus, J., 2000. Non-functional requirements in software engineering. Kluwer Academic Publishing, Norwell, Massachusetts, USA.

de Miguel, M. A., F. Briones, J., Silva Gallino, J. P., Alonso, A., Jun. 2008. Integration of safety analysis in model-driven software development. IET Software 2 (3), 260–280.

Didonet del Fabro, M., Bézivin, J., Jouault, F., 2005. AMW: a generic model weaver. En: Proceedings of the Using metamodels to support MDD Workshop, 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2005).

Dodd, J., Allen, P., Butler, J., Olding, S., Veryard, R., Wilkes, L., 2007. Cbdisae meta model for soa version 2. Tech. rep., Everware-CBDI.
URL: http://www.cbdiforum.com/public/meta_model_v2.php

Eby, M., Apr. 2007. Integrating Security Modeling into Embedded System Design. Masterthesis, Vanderbilt University.
URL: http://etd.library.vanderbilt.edu/available/etd-04022007-092035/

Elrad, T., Aldawud, O., Bader, A., 2002. Aspect-Oriented Modeling: Bridging the Gap between Implementation and Design. En: Batory, D., Consel, C., Taha, W. (Eds.), Generative Programming and Component Engineering. Vol. 2487 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 189–201.

Guinard, D., Ion, I., Mayer, S., 2011. In search of an internet of things service architecture: Rest or ws-*? a developers' perspective. En: Puiatti, A., Gu, T. (Eds.), MobiQuitous. Vol. 104 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, pp. 326–337.

Hernandez, V., Lopez, L., Prieto, O., Martinez, J. F., Garcia, A. B., Silva, A. D., 2009. SOA en automatización de pymes manufactureras. 2009 Third International Conference on Emerging Security Information Systems and Technologies, 87–92.

Illner, S., Krumm, H., Lück, I., Pohl, A., Bobek, A., Bohn, H., Golatowski, F., 2006. Model-based management of embedded service systems - an applied approach. En: AINA (2). IEEE Computer Society, pp. 519–523.

Illner, S., Pohl, A., Krumm, H., nov. 2005. Model-driven security management of embedded service systems. En: Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE. p. 6 pp.
DOI: 10.1109/IECON.2005.1569326

ISO/IEC, 2011. ISO/IEC 25010 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. ISO, Geneva, Switzerland.

Kim, A., Luo, J., Kang, M., 2007. Security Ontology to Facilitate Web Service Description and Discovery. En: Journal on Data Semantics IX. Vol. 4601 of Lecture Notes in Computer Science. Springer Berlin, pp. 167–195.

Langer, P., Wieland, K., Wimmer, M., Cabot, J., 2011. From uml profiles to emf profiles and beyond. En: Bishop, J., Vallecillo, A. (Eds.), Objects, Models, Components, Patterns. Vol. 6705 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 52–67.

Meiko Jensen and Sven Feja, 2009. A Security Modeling Approach for Web-Service-Based Business Processes. En: 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, ECBS 2009, San Francisco, California, USA. IEEE Computer Society, pp. 340–347.

Menzel, M., Meinel, C., Sep. 2009. A Security Meta-model for Service-Oriented Architectures. En: 2009 IEEE International Conference on Services Computing. IEEE, Bangalore, India, pp. 251–259.
DOI: 10.1109/SCC.2009.57

Microsoft, 2012a. Micro Framework Web Page.
URL: http://www.microsoft.com/en-us/netmf/default.aspx

Microsoft, 2012b. WSDAPI.
URL: http://msdn.microsoft.com/en-us/library/windows/desktop/aa826001%28v=vs.85%29.aspx

Mouelhi, T., Fleurey, F., Baudry, B., Le Traon, Y., 2010. A model-based framework for security policy specification, deployment and testing. Model Driven Engineering Languages and Systems 5301/2010, 537–552.

Nabil, S., Mohamed, B., 2012. Security ontology for semantic scada. En: Malki, M., Benbernou, S., Benslimane, S. M., Lehireche, A. (Eds.), ICWIT. Vol. 867 of CEUR Workshop Proceedings. CEUR-WS.org, pp. 179–192.

OASIS, 2006. Web services security: Soap message security 1.1 (ws-security 2004). Security 2003 (February), 76.
URL: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

OASIS, 2009. Devices Profile for Web Services Version 1.1. OASIS (July).
URL: http://docs.oasis-open.org/ws-dd/dpws/1.1/pr-01/wsdd-dpws-1.1-spec-pr-01.html

OMG, 2007. Specification. A UML Profile for MARTE.

OMG, 2008. UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms Version 1.1.

OMG, 2009. Service oriented architecture Modeling Language (SoaML)-Specification for the UML Profile and Metamodel for Services (UPMS).

OMG, 2011. Business Process Model and Notation (BPMN).
DOI: 10.1007/s11576-008-0096-z

Ortiz, G., Hernández, J., 2006. Service-oriented model-driven development: Filling the extra-functional property gap. Service-Oriented Computing–ICSOC 2006 4294/2006, 471–476.

Satoh, F., Nakamura, Y., Mukhi, N., Tatsubori, M., Ono, K., 2008. Methodology and Tools for End-to-End SOA Security Configurations. En: 2008 IEEE Congress on Services, SERVICES I. IEEE Computer Society, Honolulu, Hawaii, USA, pp. 307–314.

Shopov, M., Matev, H., Spasov, G., 2007. Evaluation of Web Services Implementation for ARM-based Embedded System. En: Proceedings of ELECTRONICS'07. Sozopol, Bulgaria, pp. 79–84.

Silva Gallino, J. P., de Miguel, M. A., Briones, J. F., Alonso, A., 2010. Model-Driven Development of a Web Service-Oriented Architecture and Security Policies. En: 2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing. IEEE Computer Society, Los Alamitos, CA, USA, Carmona, Spain, pp. 92–96.

Silva Gallino, J. P., de Miguel, M. A., Briones, J. F., Alonso, A., 2011b. Domain-Specific Multi-Modeling of Security Concerns in Service-Oriented Architectures. LNCS - 8th International Workshop on Web Services and Formal Methods, WS-FM'11.

Silva Gallino, J. P. and de Miguel, M. A. and Briones, J. F. and Alonso, A., 2011a. Multi Domain-Specific Modeling of the Security Concerns of Service-Oriented Architectures. Services Computing, IEEE International Conference on 0, 761–762.
DOI: 10.1109/SCC.2011.102

SOA4D, 2007. Web Page.
URL: https://forge.soa4d.org/

Tarr, P., Ossher, H., Harrison, W., Sutton Jr., S. M., 1999. N degrees of separation: multi-dimensional separation of concerns. International Conference on Software Engineering, 107 – 119.

Theorin, A., Ollinger, L., Johnsson, C., May 2012. Service-oriented process control with grafchart and the devices profile for web services. En: 14th IFAC Symposium on Information Control Problems in Manufacturing (INCOM). Bucharest, Romania.

Unger, S., Pfeiffer, S., Timmermann, D., may 2012. Dethroning transport layer security in the embedded world. En: New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on. pp. 1 –5.
DOI: 10.1109/NTMS.2012.6208685

Wada, H., Suzuki, J., Oba, K., 2008. Early Aspects for Non-Functional Properties in Service Oriented Business Processes. Services, IEEE Congress on 0, 231–238.
DOI: 10.1109/SERVICES-1.2008.76

WS4D, 2007. Web Page.
URL: http://www.ws4d.org/